Design and Evaluation of a Reconfigurable ECU Architecture for Secure and Dependable Automotive CPS

Bikash Poudel and Arslan Munir, *Senior Member, IEEE*

Abstract— The next generation of automobiles integrate a multitude of electronic control units (ECUs) to implement various automotive control and infotainment applications. However, recent works have demonstrated that these pervasively computerized modern automobiles are susceptible to security attacks that could compromise the physical safety of the driver and/or passengers. In this paper, we propose a novel ECU architecture for automotive cyber-physical systems (CPS) that simultaneously integrates both security and dependability primitives in the design with negligible performance, energy, and resources overhead. We implement our proposed ECU architecture on Xilinx Automotive (XA) Spartan-6 FPGA. We demonstrate the effectiveness of our proposed architecture using a steer-by-wire (SBW) application over controller area network (CAN) with flexible data rate (CAN FD) as a case study. We also optimize and implement a prior secure and dependable automotive work on NXP quad-core iMX6Q SABRE automotive board. We quantify the performance, energy, and error resilience of our proposed architecture for the SBW case study. Results reveal that our proposed architecture can attain a speedup of 47.9$\times$ while consuming 2.4$\times$ lesser energy than the optimized SABRE board implementation of security and dependability primitives. We further perform a comparative analysis of prior designs and the proposed ECU architecture for different in-vehicle networks, viz., CAN, CAN FD, and FlexRay. Results verify the feasibility as well as the superiority of the proposed ECU over other prior designs in terms of response time, energy efficiency, and error resilience.

Index Terms—Automotive, cyber-physical systems, security, dependability, ECU, multicore, reconfigurable architectures, FPGA, steer-by-wire

1 Introduction and Motivation

Modern automobiles (also known as cybercars) are intricate distributed cyber-physical systems (CPS) comprising of more than hundred heterogeneous processors, numerous radio interfaces, hundreds of megabytes of complex embedded software, and multiple in-vehicle networks and protocols. The controller area network (CAN) is the most prevalent protocol for communication among ECUs in automotive CPS. To enhance energy efficiency, automated control, and user comfort, modern automobiles are forsaking traditional mechanical and hydraulic subsystems in favour of x-by-wire subsystems. However, x-by-wire systems have stringent real-time performance and reliability requirements, which pose significant challenges for implementation over bandwidth-limited CAN. FlexRay with high speed data transfer and inherent fault tolerance features is a suitable replacement for CAN protocol. Nevertheless, the automotive industry is reluctant to adopt the FlexRay communication protocol because the transition requires a major revamp of automotive electronic subsystems. CAN with flexible data-rate (CAN FD) bridges the gap between CAN and FlexRay, and provides an easier alternative for implementing new automotive control applications, such as x-by-wire.

Realization of next generation automotive CPS applications, such as x-by-wire, requires incorporating dependability and security features in automotive ECUs and in-vehicle networks. The automotive CPS applications have strict dependability requirements as dictated by International Organization for Standardization (ISO) 26262 [1]. The adherence to the ISO 26262 standard requires that at least one critical fault must be tolerated by the automotive applications without loss of functionality. Meeting these dependability requirements presents various challenges. Automobiles have to endure harsh operational environments (including external noise and radiations) that render electronic systems vulnerable to *permanent*, *transient*, and *intermittent* faults. Permanent faults could impair or stop the correct functionality of the system while transient faults induce soft errors in the system. The intermittent faults oscillate between quiescent and active states [2], that is, the component functions correctly when the fault is quiescent and the component malfunctions when the fault is active. A loose electrical connection is an example of intermittent faults. Furthermore, in-vehicle distributed control systems are traditionally designed without security in mind. All of the current in-vehicle communication protocols, such as CAN, CAN FD, and FlexRay, carry messages in plaintext format, which could be read and altered by any device connected to the automotive bus. These security threats are further aggravated by the increasing integration of automotive systems with external entities, such as consumer electronics, other vehicles, and wireless networks.

The cardinal challenge in automotive CPS design is to integrate security and dependability simultaneously while ensuring that hard real-time constraints of the automotive

- *Bikash Poudel is currently working at Intel Corporation. Arslan Munir is with the Department of Computer Science, Kansas State University, Manhattan, Kansas.*
E-mail: {bpoudel@ksu.edu; amunir@ksu.edu}

CPS applications are not violated. This paper addresses this cardinal challenge of automotive CPS design while also minimizing energy consumption. Temporal performance (i.e., meeting timing constraints) is often considered as a system's quality of service (QoS) measure. We assert that the system's QoS must also be construed as a dependability measure that can impact the system's availability and safety beyond a certain *critical threshold* as the driver can totally lose the control of his/her vehicle beyond that critical threshold [3]. This critical threshold also defines the notion of *behavioural reliability*, which is defined as the probability that the system's worst-case response time is less than the critical threshold. The behavioral reliability measure can be used to ensure the stability of automotive CPS design [4].

The security susceptibilities and safety requirements of automotive CPS justify inclusion of security and dependability primitives in the design of automotive CPS. Earlier works have addressed some of the security and safety issues of in-vehicle distributed systems, vehicle-to-infrastructure communication, and vehicle-to-vehicle communication [4] [5] [6] [3]. Prior work by Munir and Koushanfar [3] is the most relevant work to this paper, in which the authors proposed a secure and dependable approach for automotive systems, and presented a primitive implementation of the proposed approach on an Intel processor. The work by Munir and Koushanfar [5] integrated security primitives by software implementation of encryption and authentication protocols. Furthermore, the work did not implement the proposed approach on an automotive ECU. To overcome the limitations of earlier work, this work integrates security and dependability primitives in hardware to attain better performance and energy efficiency than software-based approaches. In this work, we propose a novel secure and dependable automotive ECU architecture and compare it with prior multicore-based ECU architectures [5] [3] with respect to temporal performance, energy efficiency, and error resilience. Our main contributions are as follows:

- We summarize various security and dependability standards and specifications for automotive ECUs. We also provide an overview of security and dependability features in contemporary ECUs.
- We enhance the security structure of a prior secure and dependable automotive approach (Munir and Koushanfar [5]), which we refer to as "baseline design" (BD). We reinforce the security of the BD by replacing secure hash algorithm-2 (SHA-2) based hash-based message authentication code (HMAC) with SHA-3 based HMAC. We further optimize and implement the BD on the NXP iMX6Q SABRE automotive board. We refer to this optimized BD implementation as OBD. Furthermore, we quantify the error resilience of the OBD approach and evaluate the interplay of performance, security, and fault tolerance for the automotive SBW subsystem.
- We propose a novel ECU architecture that incorporates security and dependability primitives while minimizing energy consumption and ensuring that real-time constraints of automotive CPS applications are satisfied. The proposed architecture is suitable for a broad range of automotive subsystems, such as x-by-wire, infotainment, and powertrain, etc.
- We implement our proposed secure and dependable ECU architecture on Xilinx Automotive (XA) Spartan-6 field-programmable gate array (FPGA). We refer to this FPGA implementation of our proposed ECU architecture as EAF.
- We analyze our proposed ECU architecture and approach using a SBW application over CAN FD as a case study. We compare the temporal performance, energy efficiency, and error resilience of our proposed ECU architecture with comparable security and dependability primitives' implementation on the existing state-of-the-art ECU architectures, such as NXP iMX6Q SABRE automotive board.
- We perform a comparative analysis of the proposed approaches (BD, OBD, and EAF) for different in-vehicle networks, viz., CAN, CAN FD, and FlexRay, in terms of pure delay (pure delay comprises of computation time required for processing control algorithm and security and dependability primitives, plus the transmission time) and response time. Results verify the feasibility of all the proposed approaches (BD, OBD, and EAF) as well as the superiority of EAF over BD and OBD in terms of pure delay and response time.
- We highlight the future research directions for designing secure and dependable automotive CPS.

The rest of this paper is organized as follows. Section 2 discusses prior research efforts in the field of secure and dependable automotive CPS design. Section 3 elaborates on the contemporary standards and specifications for automotive electronics, which have been adopted in state-of-the-art automotive grade microprocessor and microcontrollers. Section 4 explains our proposed security and dependability approach. Furthermore, this section illustrates the design modification and optimizations incorporated in OBD as compared to BD. In Section 5, we propose our novel ECU architecture that simultaneously integrates security and dependability features in automotive CPS without violating the real-time constraints inherent in automotive CPS applications. Section 6 discusses the SBW system case study that we have used to demonstrate the effectiveness of our proposed ECU architecture. Section 7 details the experimental setup and evaluation results demonstrating the performance, energy, and error resilience features as well as the feasibility of our proposed approach and the proposed ECU. Section 8 highlights future research directions for designing secure and dependable automotive CPS. Finally, Section 9 concludes this work.

2 Related Work

Security for automotive CPS has been studied in previous works. Koscher et al. [7] and Checkoway et at. [8] analyzed the internal and external attack surfaces of a modern automobile through which an attacker could control automotive subsystems (e.g., engine, brakes, windshield wipers) while completely ignoring the driver's input. The authors discovered a broad range of attack vectors that included cellular radio, bluetooth, CD player, and mechanics tools. One interesting work that incorporates security features such as CAN message confidentiality and integrity is by Chavez et al. [9]. Chavez et al. [9] suggested using the security services, viz., confidentiality, integrity, authentication, nonrepudiation, and access control, of the OSI (Open Systems Interconnection) reference model (ISO 7498-2 [10]) for securing CAN protocol. Following this, the authors proposed that access control could be taken care of at higher layers in the protocol, that integrity could be enforced by using hash algorithms, and that confidentiality could be enforced by using RC4 encryption of CAN frames. However, the foible of this work is that the remaining two OSI services, viz., authentication and non-repudiation, were not considered.

Various prior works [11], [12], [13], [14] discussed incorporation of message authentication codes (MACs) in CAN data frames to secure in-vehicle communication. Nilsson et al. [13] proposed to use a compound MAC added to the payload of the CAN frames, which could be used to detect and possibly recover from injection and modification attacks in the in-vehicle networks. In their proposed scheme, the authors calculated the MAC over four consecutive CAN messages and the resulting MAC was partitioned into four 16-bit blocks and transmitted in the cyclic redundancy code (CRC) field of the next four CAN messages. The protocol required a total of eight CAN messages for MAC verification, and thus resulting in delayed verification of data integrity and data authentication. Furthermore, the protocol did not provide protection against replay attacks nor was able to identify individual faulty messages in case of MAC verification failure.

Several previous works [11], [12], [14] proposed security mechanisms based on MACs and counters for CAN to prevent both masquerade attacks and replay attacks. However, these works did not consider message confidentiality nor the fault tolerance (FT) aspects of messages transmitted over the CAN bus. Herrewege et al. proposed CANAuth [14]—a simple and backward compatible broadcast authentication protocol for CAN bus. The scheme provided two main security services for the in-vehicle CAN network: CAN message authentication and resistance to replay attacks. The authentication was achieved by using HMAC and the resistance to reply attacks was achieved by augmenting a counter with the message. To make the proposed scheme backward compatible with the existing CAN bus, CANAuth used out-of-band channel to send authentication data using CAN+ protocol [15]. The drawback of the approach, however, is that the scheme did not provide message confidentiality and source authentication. Furthermore, the protocol assumed that all of the CANAuth nodes possessed a preshared key.

Groza et al. [11] proposed LiBrA-CAN, which provided source authentication and resilience to replay attacks for CAN. The protocol was based on symmetric primitives and utilized two interesting procedures that the authors referred as key splitting and MAC mixing. Instead of performing source node authentication independently for each node, the approach split authentication keys between groups of multiple nodes which led to an efficient progressive authentication. The weakness of the protocol is that the message confidentiality was not embodied in the CAN network which left the CAN network vulnerable for traffic analysis related attacks. Wolf et al. [16] presented a vehicular hardware security module (HSM) that was implemented in Xilinx Virtex-5 FPGA to secure in-vehicle ECUs and their communication. However, the HSM did not incorporate any FT features. Furthermore, the work did not evaluate the interplay of performance, energy, and FT which is essential for secure and dependable automotive CPS applications.

Several earlier work explored dependability for automotive embedded systems. Beckschulze et al. [17] investigated FT approaches based on dual-core microcontrollers. The work compared the functional monitoring architectures that monitored the hardware executing the application. Baleani et al. [18] studied various FT architectures for automotive including lock-step dual processor architecture, loosely-synchronized dual processor architecture, and triple modular redundant architecture. The work, however, did not quantify the architectures' FT capabilities subject to real-time constraints of automotive CPS. Rebaudengo et al. [19] investigated sofware-based FT techniques for soft error detection. The authors presented an approach for detection of soft errors by automatically introducing data and code redundancy into an existing program written in a high-level language. The proposed approach, however, resulted in an average performance penalty of 5×, and thus may not be feasible for automotive CPS with hard real-time constraints.

In summary, although previous works identified security vulnerabilities in automotive systems, these works did not present an integrated approach for designing secure and dependable automotive CPS. Munir and Koushanfar [5] presented a multi-core ECU based design of secure and dependable automotive embedded systems using SBW application as a case study. However, the work did not implement the proposed approach on an automotive-grade processor. In this paper, we enhance the security structure of the secure and dependable approach proposed in [5], which we refer to as the "baseline design", and further optimize and implement the baseline design on an automotive-grade processor.

3 Security and Dependability Standards and Specifications for Automotive ECUs

Contemporary automobiles utilize a variety of microcontroller/microprocessor units (MCUs/MPUs) as ECUs to implement various control and infotainment applications. Recognizing the need for incorporation of security and dependability primitives in automotive ECUs, automotive original equipment manufacturers (OEMs), suppliers, and standards organizations have developed a few specifications and standards for automotive ECUs. In this section, we discuss the contemporary standards and specifications for automotive ECUs that have been deployed in state-of-the-art automotive ECUs.

3.1 Security

A number of specification activities and security module architectures have reached sufficient maturity to be accepted as a standard within the automotive engineering community, while some of the specifications are on process to be standardized. We discuss some of these security specifications for automotive ECUs in the following.

Secure Hardware Extension (SHE): SHE [20] is one of the security specifications that has now been accepted as an open and free standard. SHE was launched by Hersteller Initiative Software (HIS), which is a working group of all German car manufacturers. SHE is an industrial standard that describes a hardware extension for adding essential security functionalities, such as hardware cryptographic module, secure boot, management of security keys, etc. SHE is implemented as an on-chip extension to microcontroller. The cryptographic service engine (CSE) modules used in many automotive MPUs and MCUs including NXP MPC564xB/C implement the security functions described in SHE functional specification.

Hardware Security Modules (HSMs): The EVITA [21] (E-safety Vehicle Intrusion proTected Applications) is a European Union funded project that has developed a set of guidelines that detail the design, verification, and prototyping of security architectures for automotive ECUs. EVITA has developed three HSMs: full, medium, and light, for different automotive use cases. These HSM are used as an extension to the existing automotive MCUs and MPUs to provide security for in-vehicle networks. Furthermore, PRESERVE [22] has extended the EVITA project to develop, implement and test a scalable security subsystem for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) applications. The security subsystem identifies and addresses threats in V2X (i.e., V2V and V2I) communication in future intelligent transportation system (ITS). PRESERVE has designed an integrated vehicle security architecture (VSA) to protect data and control information during the V2X communication between ITS stations.

Trusted Platform Module (TPM): Trusted computing group (TCG) [23] is a computer industry consortium that released TPM specification. The TPM specification was standardized by ISO and International Electrotechnical Commission (IEC) in 2009 and was published as ISO/IEC 11889. The TPM supports secure keys for authentication and encryption functions. The TPM handles cryptographic operations such as symmetric/asymmetric key generation, symmetric/asymmetric encryption/decryption, hashing, and random number generation. The TPM is usually a microcontroller that securely stores passwords, digital keys, and certificates that can provide unique identification. The TPM is implemented either as an external peripheral with a communication bus to automotive MCU/MPU or as an embedded portion of another integrated circuit (IC), such as Ethernet controller.

ARM's TrustZone: ARM's TrustZone [24] is a proprietary specification/guideline to support the development of secure and safe embedded systems. ARM's TrustZone [24] enables system-wide security by incorporating protective measures into the ARM processor, bus fabric, and system peripheral intellectual property (IP). The TrustZone provides isolation between the normal world (operating system and application layer) and the secure world, and thus can be used for sensitive operations such as cryptographic operations, key management, and integrity checking.

3.2 Dependability

ISO 26262 [1] is a functional safety standard developed for road vehicles. We clarify that functional safety is a subset of system dependability. ISO 26262 has introduced automotive safety integrity levels (ASILs) as a risk classification scheme. ASILs are modification of safety integrity levels (SILs) used in IEC 61508 for the automotive industry. ASIL defines safety requirements that must be implemented in the design and development phases of the system such that the system provides sufficient margin of safety for users (driver, passenger, etc.) even in conditions of failure. ASIL is established by careful inspection of hazards and risks, and detailed analysis of severity, exposure, and controllability of the vehicle's operational scenario. An ASIL is specified as one of the four levels: ASIL-A, ASIL-B, ASIL-C, and ASIL-D, where ASIL-D dictates the highest integrity requirements on the product/function while ASIL-A stipulates the lowest. Most of the recent safety-critical MPUs and MCUs fall in one of the four ASIL grades. The safety and security features available in some MPUs/MCUs that are used as ECUs in the automotive system are shown in Table 1.

Automotive electronics council (AEC) is another organization that sets qualification standards for the components used in automotive industry. The AEC-Q100 [25] is the automotive standard specification developed by AEC that dictates failure mechanisms based stress tests for packaged ICs. The AEC-Q100 labels the qualified products (ICs) in terms of automotive temperate grades that signify the capability of the IC to operate correctly within a specified temperature range. The automotive temperature grades specified by AEC-Q100 are [25]: grade 0 (-40°C to +150°C), grade 1 (-40°C to +125°C), grade 2 (-40°C to +105°C), grade 3 (-40°C to +85°C), and grade 4 (0°C to +70°C). Many of the automotive MPUs/MCUs satisfy AEC-Q100 qualification with a particular grade depending on the intended usage/application of the MPU/MCU. For example, grade 0 qualified products should be used for under the hood automotive applications, where ambient temperature can rise up to 105°C to 130°C or even higher. Similarly, grade 1 qualified products can be used in chassis locations not directly exposed to the heat from the engine or exhaust.

3.3 Deficiencies in Contemporary ECUs

As illustrated in Table 1, security features in MCUs/MPUs are based on SHE [20] and HSMs from EVITA [21]. The TPM [23] from the TCG is neither designed nor suitable for automotive CPS as the TPM lacks performance, robustness and cost-effectiveness desired for automotive functions. Further TPM does not provide all the required security features for automotive CPS. Although SHE and HSM provide all relevant security features, the internal architecture of SHE and HSM is not FT or dependable. An error in the operation of these modules makes automotive CPS vulnerable to security breaches and malfunctions. Hence, there is a need for developing an approach for design of automotive CPS that simultaneously integrates both security and dependability primitives. This work aims to fill in this gap in the research and development of automotive CPS and proposes a novel approach (Section 4) and an ECU architecture based on that approach (Section 5)

TABLE 1: An overview of security and dependability features in MCUs/MPUs used as ECUs in modern automobiles.

Vendor Name	Processor Name	Security Feature	Dependability Feature
NXP/Freescale	Qorivva MCUs	HSM-based security module	ISO 26262 ASIL-C,-D certified
	iMX MCU	Built-in security core	None
	Kinetis EA series	None	Supports ISO 26262/IEC 61508
	MPC57xx	None	ISO 26262 ASIL-D certified
	MPC56xx	None	ISO 26262 ASIL-D certified
	S32K144	None	ISO 26262 ASIL-B or higher certified
Texas Instruments	DRA746 Jacinto	Built-in security module	None
	Hercules TMS570	Advanced JTAG security module	ISO 26262 ASIL-D/IEC 61508 SIL-3 certified
	Hercules TMS470M	None	ISO 26262 ASIL-D certified
	TDAx SoC	None	ISO 26262 ASIL-B certified
STMicroelectronics	SPC5 Series	SHE and EVITA based security module	ISO 26262 ASIL-D,-B certified
Fujitsu	'Atlas' MB9DF126	SHE based security module	None
Qualcomm	Snapdragon	Qualcomm Haven security platform	None
Atmel	ATSAMV70x/71x	Built-in security module	AEC Q100 Grade-2 qualification
	ATSAMDA1x	None	AEC Q100 Grade-2 qualification

that simultaneously integrates security and dependability features in automobiles.

4 Secure and Dependable Approach for Automotive CPS

Automotive embedded systems have various design challenges including resource limitation (e.g., memory, processing, bandwidth) and applications' real-time deadlines. Implementation of security and dependability primitives and protocol under these constraints provide limited freedom for the designer. Bounded by these constraints, we have proposed and designed a novel ECU architecture in this work that leverages an enhanced version of a prior secure and dependable automotive approach (by Munir and Koushanfar [5]), which we refer to as "BD". The BD considers CAN as the in-vehicle communication protocol whereas our enhanced approach (OBD) considers CAN FD, which is more amenable for safety-critical automotive CPS applications. Fig. 1 depicts our proposed secure and dependable approach for automotive CPS design. The figure shows the operations involved at both sending and receiving nodes to incorporate security and dependability primitives. This section first presents the security threat model against which our proposed approach provides resilience. We then elaborate the security and dependability features provided by BD and OBD with a brief comparison between these two modus operandi. Finally, we discuss the techniques that we have used to optimize the BD code to generate the OBD code.

4.1 Security Threat Model

To better illustrate our proposed approach and the resilience it provides against security vulnerabilities, we characterize the capabilities of an adversary aiming to infiltrate automobile's internal network (e.g., CAN, CAN FD, FlexRay) to carry out numerous attacks. Modern automobiles provide a variety of attack surfaces, ranging from mechanical tools to on-board diagnostics ports (OBD-II) and from CD players to various short/long range wireless interfaces (e.g., bluetooth, remote keyless entry, wireless tire pressure sensors, telematics systems, global positioning systems, satellite radio, digital radio), which make automotive systems vulnerable to a broad range of attacks [8]. For example, an adversary who has infiltrated an automotive braking system may cause the driver to completely lose control of his/her vehicle. Thus, in order to ensure security and safety of passengers and vehicles, automotive CPS needs to integrate security and dependability primitives in the design, in particular, confidentiality, integrity, and authentication [3]. Assuming that an adversary has gained access to the automotive internal network, this section summarizes the threat model against which our proposed approach provides resilience.

Threat 1—Passive Eavesdropping & Traffic Analysis $\rightarrow$ Need for Confidentiality: Passive attacks come in two flavours: passive eavesdropping or monitoring and traffic analysis. In passive eavesdropping, an adversary can sniff and store transmission of messages from one ECU to another over in-vehicle networks. If the transmitted messages are not encrypted, then an adversary can easily extract information from the eavesdropped messages to launch further attacks. The attacker can further garner additional valuable information by performing traffic analysis on the eavesdropped messages. For instance, for the x-by-wire systems, if an adversary is aware of the initial location of the vehicle, then, by eavesdropping and traffic analysis of the steering angle, accelerator, and braking values, the adversary can be able to trace the car which may put the passengers and driver of the vehicle at risk [3].

If the transmitted messages on an in-vehicle network are secured by some encryption mechanism, it makes traffic analysis attacks difficult to perform albeit an adversary can still obtain partial or complete information from the messages. Moreover, an adversary can have the ability to request generation of encrypted messages from ECUs and thus knowledge of the plain-text can be used to determine the encryption key, decrypt complete packets, or obtain other valuable information through traffic analysis. The traffic analysis on encrypted messages can enable an attacker to determine the location and identity of communicating hosts (ECUs in car) and to observe the frequency and length of messages being exchanged that can provide insights into the function implemented by the ECU (e.g., transmission control). Nevertheless, encryption of messages greatly limit traffic analysis attacks. Thus, it is imperative for OEMs to provide confidentiality of messages and data over in-vehicle networks to safeguard operational

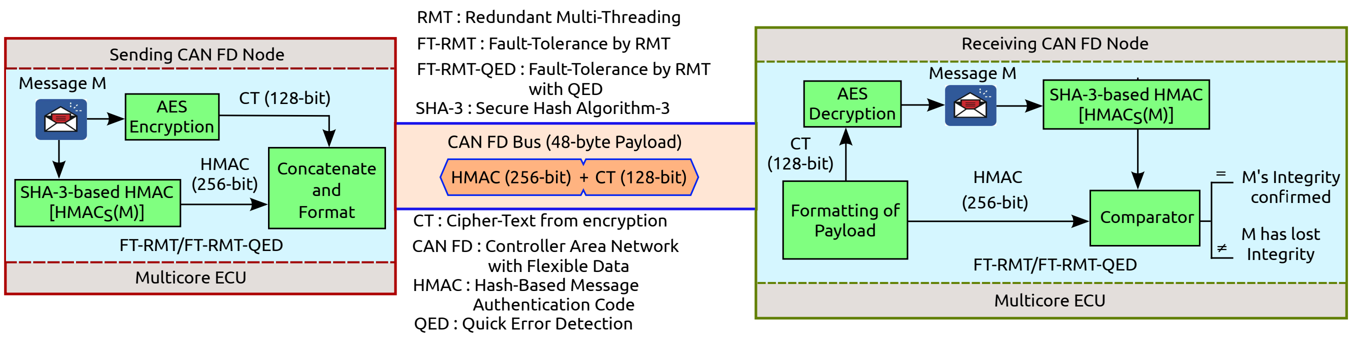
Sending CAN FD Node
Message M
AES Encryption
CT (128-bit)
SHA-3-based HMAC [HMACS(M)]
HMAC (256-bit)
Concatenate and Format
FT-RMT/FT-RMT-QED
Multicore ECU
RMT : Redundant Multi-Threading
FT-RMT : Fault-Tolerance by RMT
FT-RMT-QED : Fault-Tolerance by RMT with QED
SHA-3 : Secure Hash Algorithm-3
CAN FD Bus (48-byte Payload)
HMAC (256-bit) + CT (128-bit)
CT : Cipher-Text from encryption
CAN FD : Controller Area Network with Flexible Data
HMAC : Hash-Based Message Authentication Code
QED : Quick Error Detection
Receiving CAN FD Node
AES Decryption
Message M
SHA-3-based HMAC [HMACS(M)]
CT (128-bit)
HMAC (256-bit)
Formatting of Payload
Comparator
=
M's Integrity confirmed
≠
M has lost Integrity
FT-RMT/FT-RMT-QED
Multicore ECU

Fig. 1: Secure and Dependable Approach for Automotive CPS Design.

security, privacy, and consumer trust.

Threat 2—Active Eavesdropping & Message Injection → Need for authentication and integrity: Active attacks involve active eavesdropping where an adversary can modify the message content or create a false stream of messages. Active attacks can be further classified into four categories: masquerade attack, replay attack, modification of messages, and denial of service. A *masquerade attack* takes place when an unauthorized entity impersonates an authorized entity in the network. This can happen when an unauthorized entity captures authentication sequences and then replay the captured authentication responses after some time to gain authentication in the network. In the automotive CPS scenario, the attacker can cause a malicious ECU to masquerade as a genuine ECU to gain access to the network and participate in communication with other ECUs. A *replay* attack involves the passive capture of the messages and its subsequent retransmission either to gain unauthorized access or cause abnormal operation of automotive CPS, which may result in vehicle crash. In *message modification attack*, the adversary modifies the message(s) transmitted on in-vehicle network and can take various forms, such as altering a legitimate message, injecting a malicious message, delaying the transmission of messages, and/or reordering the transmission of messages, to produce a malicious effect. Finally, *denial of service attack* inhibits the normal usage or management of an in-vehicle network by authorized entities. An attacker can accomplish the denial of service attack by various means, such as overwhelming the in-vehicle network with malicious messages, compromising the vehicle ECUs, etc.

In automotive CPS, an adversary can launch active attacks by various means such as by attaching his/her own device or compromising a valid user device (e.g., a cell phone attached to the infotainment system) in order to transmit fraudulent (or malicious) requests (commands, codes, or messages) in the system [3]. Similarly, malicious messages can be injected by the adversary by encoding the messages on a CD as a song or video file and then manipulating the user to play that song by social engineering.

Since majority of the contemporary automotive systems do not incorporate the security attributes, *threat 1* and *threat 2* are possible. The vulnerabilities caused by these threats include infringement on confidentiality, integrity, and authentication of the messages disseminated on in-vehicle networks. By exploiting theses vulnerabilities, an adversary can potentially be able to circumvent most of the safety-critical automotive systems while completely ignoring the driver's input.

4.2 Security

To countermeasure the possible attacks mentioned in Section 4.1, our proposed approach (Fig. 1) integrates confidentiality, integrity, and authentication in automotive CPS with CAN FD as the vehicular network. However, our proposed OBD approach, as shown in Fig. 1, is "encrypt-and-MAC" as compared to "MAC-then-encrypt" approach of BD [5]. Encrypt-and-MAC provides comparable security to MAC-then-encrypt, however, encrypt-and-MAC has lesser computation time and computation overhead than MAC-then-encrypt. The BD [5] uses 128-bit advanced encryption standard (AES-128) for integrating confidentiality and SHA-2/SHA-256 based HMAC for integrating message integrity and authentication. Our approach reinforces the BD security by replacing SHA-2 based HMAC with SHA-3 based HMAC. Furthermore, the OBD is made resistant to masquerade and replay attacks by embedding a 64-bit counter value to the original message. The input to the AES and the HMAC module is 128-bit message where the first 64-bit is the original message and the second 64-bit is the counter value.

Our proposed OBD approach (Fig. 1) requires only one HMAC computation and one AES-128 encryption of the original message as compared to three AES-128 encryptions and one HMAC computation in the BD. In our "encrypt-and-MAC" approach, the HMAC of the message is not encrypted because the HMAC is collision resistant, and is computed with a secure secret key that is only known to the legitimate sender and the receiver nodes. The output of the HMAC computation (256-bit *message digest*) and the AES-128 encryption (128-bit *ciphertext*) are concatenated to generate a 384-bit (48-bytes) message. The sending node then transmit the 384-bit concatenated message to the receiver node via the CAN FD bus. The payload size of the CAN FD bus is 64 bytes so it is able to transfer 48 bytes of the message in one cycle. Hence, our enhanced approach saves two AES-128 computations at the sender node and speeds up CAN data transfer by 6× as the BD approach using CAN required six CAN messages to transmit the encrypted message and hash (the maximum payload size of the CAN message is 8 bytes [26]). We clarify that the data transfer speedup of OBD over BD is due to the larger payload size (64 bytes) of CAN FD as compared to the CAN payload size of 8 bytes although the combined encrypted and MAC-ed message size for both BD and OBD is 384 bits (48 bytes).

Eq. (1) summarizes the time required by the operations at the sender CAN FD node to incorporate the security primitives for the message.

$$\mathcal{T}^{\mathcal{S}}_{(\mathcal{M}\|\mathcal{C})} = \mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]} + \mathcal{T}_{E[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_2]} + \mathcal{T}_{format}, \quad (1)$$

where $\mathcal{C}$ represents the 64-bit counter, $\mathcal{T}^{\mathcal{S}}_{(\mathcal{M}\|\mathcal{C})}$ represents the time required at the sender CAN FD node to add security primitives to the message concatenated with the counter $(\mathcal{M}\|\mathcal{C})$, $\mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}$ denotes the time needed to compute the HMAC of the message concatenated with the counter $(\mathcal{M}\|\mathcal{C})$ using secret key $\mathcal{K}_1$, $\mathcal{T}_{E[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_2]}$ designates the time required to compute AES encryption of the message concatenated with the counter $(\mathcal{M}\|\mathcal{C})$ using secret key $\mathcal{K}_2$, and $\mathcal{T}_{format}$ symbolizes the time needed to concatenate and format the sending CAN FD node message.

Eq. (2) captures the time required at the receiver CAN FD node to regenerate the original message and to check the integrity of the received message.

$$\mathcal{T}^{\mathcal{R}}_{(\mathcal{CT}\|\mathcal{MAC})} = \mathcal{T}_{format} + \mathcal{T}_{D[\mathcal{CT},\mathcal{K}_2]} + \mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]} + \mathcal{T}_{cmp}, \quad (2)$$

where $\mathcal{T}^{\mathcal{R}}_{(\mathcal{CT}\|\mathcal{MAC})}$ designates the time required to recover the original message from the received message at the receiver node, $\mathcal{T}_{format}$ symbolizes the time needed to format the received CAN FD payload, $\mathcal{T}_{D[\mathcal{CT},\mathcal{K}_2]}$ denotes the time required to perform AES decryption of the ciphertext $\mathcal{CT}$ using secret key $\mathcal{K}_2$, $\mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}$ represents the time needed to compute HMAC of the decrypted ciphertext concatenated with the counter $\mathcal{CT}$ (or $(\mathcal{M}\|\mathcal{C})$) using secret key $\mathcal{K}_1$, and $\mathcal{T}_{cmp}$ designates the time required to compare the HMAC calculated from the decrypted ciphertext and the received HMAC from the sender.

4.3 Dependability

Both the BD and OBD uses FT to provide dependability to the automotive CPS. The scope of our FT approach is confined to error detection and correction for dependability. The BD and OBD uses two types of FT: FT by redundant multi-threading (FT-RMT) and FT by redundant multi-threading and quick error detection (FT-RMT-QED). The FT-RMT approach leverages RMT on a dual-core architecture. The RMT uses two threads, a master thread and a normal thread, to execute the same (safety-critical) computation. At the end of the computation, the results obtained from the two threads are compared to detect an error in the calculation. If there is an error then recomputation is carried out on both the threads to expunge the computation error. Most of the time recomputation rectifies errors because soft errors, which are caused by transient faults, constitute a majority of errors in the computation. The FT-RMT-QED enhances FT-RMT with QED [27] by inserting check instructions at different points in the master thread. When an error is detected earlier by FT-RMT-QED rather than at the end of the computation, the erroneous computation is aborted, and the computation is restarted to obtain an error-free result [5]. This early detection of errors by FT-RMT-QED helps in better adherence to the real-time deadlines of automotive CPS in the presence of faults and permits tolerance of more soft errors by allowing more recomputations in the *slack time* determined by the real-time constraints [3]. The FT techniques employed by BD and OBD can tolerate one permanent fault and multiple soft errors (by recomputations), and therefore adhere to the safety requirement of ISO 26262 standard [1].

4.4 Baseline Design Code Optimization

Although cryptographic primitives (eg., AES, HMAC) are specified in terms of functionality, the optimum performance of these cryptographic primitives on a given architecture can only be attained by optimizing the code for the underlying architecture. In this work, we focus on the optimization of cryptographic primitives on ARM cores (specifically automotive grade), which are widely used in embedded industry.

Several efficient software implementations of AES-128 and SHA-3 are available in literature [28], [29], [30], [31], [32]. We have studied these implementations and have designed efficient algorithms for AES and SHA-3 computations for our proposed OBD approach. Since our implementation is targeted for 32-bit ARM platform, we have incorporated optimization strategies such as loop-unrolling, cache-aware programming, alignment of data structures to cache line boundaries in memory, use of 32-bit data types only, and all of the efficient coding strategies mentioned in [33] in our implementation. These optimizations have enabled us to obtain a considerable speed up in the execution time of OBD as compared to the code used in BD.

5 Proposed Secure and Dependable ECU Architecture

As illustrated in Section 3, many of the contemporary MCUs/MPUs utilized as ECUs in automotive CPS do not simultaneously integrate security and dependability primitives in the design. Although SHE and HSM standards are widely being adopted by automotive industry, the internal architecture of SHE and HSM is not FT, and hence errors in the operation of these SHE and HSM based modules make automobiles vulnerable to malfunction and security breaches. In this section, we propose a novel ECU architecture that aims to alleviate the deficiencies in contemporary ECUs by simultaneously integrating security and dependability primitives in a robust and flexible manner. We first provide an overview of the proposed ECU architecture followed by the description of its operation. We then elaborate the dependability and security features incorporated by our proposed ECU architecture.

5.1 Architecture Overview

Fig. 2 depicts the overview of our proposed ECU architecture. This architecture is inspired by Xilinx Zynq-7000 system-on-chip (SoC) [34]. Our proposed ECU architecture has two programmable parts: an ARM based application processor or real-time processor (AP/RTP) and an FPGA based programmable logic fabric (PLF). The AP/RTP implements the automotive function's (e.g., SBW) control algorithms. The type of AP/RTP used depends on the automotive function and the corresponding control algorithm to be executed. The FPGA-based PLF implements cryptographic functionality (i.e., cryptographic algorithms

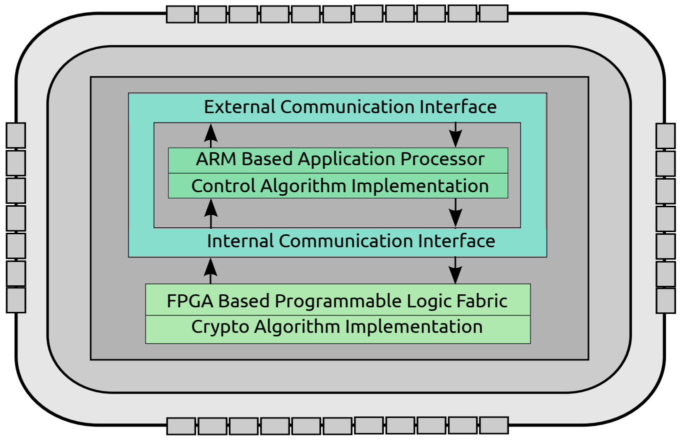
External Communication Interface
ARM Based Application Processor
Control Algorithm Implementation
Internal Communication Interface
FPGA Based Programmable Logic Fabric
Crypto Algorithm Implementation

Fig. 2: High level architecture of proposed ECU.

and protocols) for secure on-board communication and authentication. The AP/RTP and the PLF are connected by a high speed communication interface.

Our proposed ECU architecture provides a multitude of benefits. First, the PLF implements an FT cryptographic module (CM) as shown in Fig. 3. The FT is realized by using dynamic partial self-reconfiguration. Second, the PLF is a suitable platform for implementing real-time constrained compute-intensive algorithms, such as audio, video, image processing, and machine learning algorithms. The PLF can be extended to implement these compute-intensive algorithms without incurring extra cost. To add new applications on top of the CM, the designer simply needs to write a hardware descriptive language (HDL) module for the new application, and then embed this module with the CM by modifying the bitstream in the PLF. Third, the internal communication interface between the ECU's AP/RTP and PLF is inherently secure since the AP/RTP and the PLF are connected by the internal bus, which is not accessible to an attacker.

5.2 ECU Operation

At the sender node, during normal non-faulty state, the input message is read by two AES encryption and two SHA-3 based HMAC modules depicted on the left and the right part in Fig. 3. The spare modules (designated by * in figure) do not read the input message because the spare modules are disabled by the self-checking fault handler (SCFH) module. The outputs of the two AES encryption and the two SHA-3 based HMAC modules are fed to two input interfaces, which route these outputs to the comparators in triple modular redundancy (TMR). The comparators generate the comparison outputs, which are then fed to the self-checking voter (SCV). The SCV is a majority voter, which is designed based on totally self-checking (TSC) Berger code. In case of a fault, the SCV can flag itself as a faulty unit to the SCFH. The output of the SCV is then passed to the SCFH. The SCFH generates all the necessary control signals (omitted in figure for conciseness) for the operation of the FT CM. Additionally, SCFH has a small buffer that stores the results of recent computations of all the modules in the FT CM.

If there is no error in the AES and HMAC computations, then the SCFH concatenates the HMAC to the AES encrypted message and sends the concatenated message to the AP/RTP via a high speed internal bus, which then transmits the message to the receiver ECU through CAN FD bus. If there is a fault in the AES and/or HMAC computation, then the fault detection and recovery is done by Algorithm 1. The algorithm takes the input as modules in dual modular redundancy (DMR) and spare modules. The modules in DMR M_1 and M_2 are operational during non-faulty state while the spare modules M_S become operational during the faulty state. The output of the algorithm is successful reconfiguration of faulty modules.

Algorithm 1 works as follows. First, to detect an error in the computation, the outputs of the two modules in DMR are compared. If there is a mismatch in the output, recomputation is done in both the modules using previous input for which there was an error (lines $1-3$). If the recomputation yields the same output in the two modules, the *ciphertext* produced by AES ENCRYPTION module (Fig. 3) and the *message digest* produced by SHA-3 BASED HMAC module (Fig. 3) are concatenated and sent to the receiver CAN FD node. The CM returns to non-faulty mode (lines $9-10$). However, if the first recomputation does not correct the error, the algorithm tries up to NUM_SOFT_ERR recomputations (lines $4-5$), where NUM_SOFT_ERR is a threshold set on permissible error corrections by recomputations, which can be calculated based on the slack time (slack time can be determined from the real-time constraints of the application) [3]. Eq. (9) (given in Section 6.2) can help in the determination of a suitable NUM_SOFT_ERR threshold value. If recomputations fail to rectify the fault, both of the suspected faulty modules are deactivated, and the spare modules are activated for AES and HMAC computation (lines $11-13$) because it is not feasible to correct the error with DRM and further it is not possible with DMR to discern which of the two suspected faulty modules is actually faulty.

Lines $13-30$ in Algorithm 1 shows the localization and fault recovery of the faulty modules. Here, the spare modules compute AES and HMAC with the previous input and the result is routed to SCFH via the input interface. The SCFH compares this new output with the output in buffer from the previous computation to identify the faulty module. Finally, the SCFH signals the reconfiguration subsystem to partially reconfigure the faulty module. If both of the modules in the DMR configuration are faulty, then both the modules are reconfigured while the system continues operation with the spare modules until the reconfiguration is completed. However, if only one of the modules is faulty in the original DMR configuration, then the non-faulty and the spare modules start operating in DMR. The rationale for using the spare modules in the new DMR configuration is that the reconfiguration takes longer time (in order of tens of millisecond) and the CM must be functional during that reconfiguration period for fulfilling the security and dependability requirements of automotive CPS.

Similar operations are performed by the proposed ECU architecture at the receiver CAN FD node and are further explained in Section 5.3 and Section 5.4.

5.3 Dependability

The dependability requirements of automotive CPS as stipulated by ISO 26262 [1] can be met by designing a FT

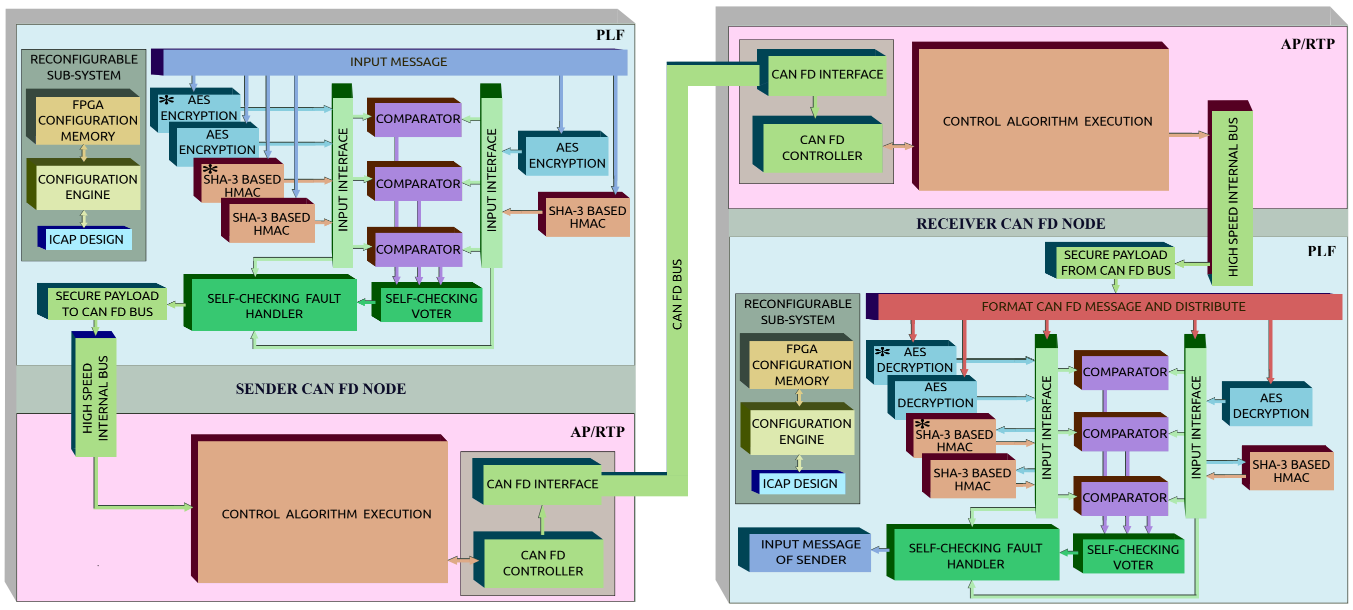

Fig. 3: Internal architecture of the secure and dependable cryptographic module (FT CM) implemented in PLF of the proposed ECU.

Algorithm 1: Fault recovery in proposed ECU.

```
Input  : modules in DMR ($M_1$ and $M_2$) and spare
         modules $M_S$
Output: reconfiguration of faulty module(s)
Data: $count = 1$, NUM_SOFT_ERR = $n_{se}$
1  if ( $Op(M_1) \neq Op(M_2)$) then
           // Op($M_i$): output of module $M_i$
2      $count \leftarrow count + 1$
3      RecomputeAES_HMAC($M_1, M_2, previousInputs$)
4      if ($count <$ NUM_SOFT_ERR) then
5          Goto Line 1
6      else
7          Goto Line 11
8  else
9      Concatenate($cipherText, messageDigest$)
10     SwitchMode(NonFaulty)
11 DeactivateModule($M_1, M_2$)
12 ActivateModule($M_S$)
13 ComputeAESHMAC($M_S$)
14 if ( $Op(M_S) \neq Op(M_1)$ && $Op(M_S) = Op(M_2)$) then
15     faultyModule $\leftarrow M_1$
16     ActivateModule($M_2$)
17     ReconfigureModule($M_1$)
18     SwitchMode(NonFaulty)
19 if ( $Op(M_S) \neq Op(M_2)$ && $Op(M_S) = Op(M_1)$) then
20     faultyModule $\leftarrow M_2$
21     ActivateModule($M_1$)
22     ReconfigureModule($M_2$)
23     SwitchMode(NonFaulty)
24 if ($Op(M_S) \neq Op(M_1)$ && $Op(M_S) \neq Op(M_2)$) then
25     faultyModule $\leftarrow M_1$ and $M_1$
26     ReconfigureModule($M_1$, $M_2$)
27     SwitchMode(NonFaulty)
```

CM. The main dependability features of the proposed FT CM in our design (Fig. 3) include:

1) DMR with extra spare modules (marked by * in Fig. 3) (one for AES and one for HMAC computation)
2) Berger code based totally self-checking combinational circuit [35]
3) Partial reconfiguration feature of Xilinx Automotive Spartan-6 FPGA [36].

The first FT attribute of our proposed system is DMR. DMR is a FT technique that uses two redundant modules to help detect the computation error(s) by comparing the outputs of the two modules. However, one potential shortcoming of DMR based FT is that the technique cannot identify (localize) the faulty module among the two operating modules. In order to resolve this identification/localization issue, our proposed ECU uses spare modules (one for AES and one for HMAC computation) which are activated only during the faulty state to detect and identify the faulty module(s). The activation of the spare modules only during the faulty state and not in the normal correct operation helps to improve energy efficiency of the automotive system. The fault detection and recovery is shown in Algorithm 1 and is thoroughly explained in Section 5.2.

The second FT feature incorporated in our proposed ECU is TSC. The TSCs are a class of circuits in which the occurrence of fault can be detected by observing the circuit output. A TSC consists of a functional circuit whose output words belong to a certain code (Berger code in our case), and a checker that monitors the output of the functional circuit to detect fault(s) in the circuit. The reliability of the circuit depends on the ability of its checker to behave correctly despite the possible occurrence of internal fault(s).

The final FT feature incorporated by our proposed ECU for fault recovery is partial reconfiguration (PR) of PLF. Our proposed ECU architecture heals the faulty modules by exploiting the PR technology as discussed in the following. Reconfigurability means the capability of programmable hardware devices, such as FPGA, to change a customized design by loading different bitstreams. A more advanced reconfiguration technology is PR where a subset of FPGA operational logic is modified by downloading a partial configuration file/bitstream. Typically, PR is achieved by overwriting the current design in the FPGA configuration memory with the partial bitstream of the new design. Xilinx FPGAs provide a dedicated internal configuration access port (ICAP) that directly interfaces to the configuration memory and accesses it. We use LogiCORE IP (Intellectual Property) XPS (Xilinx Platform Studio) HWICAP (Hardware ICAP) [37] to perform dynamic PR. The XPS HWICAP IP enables MicroBlaze processor as configuration engine to read and write the FPGA configuration memory through the ICAP at run time and perform the PR.

5.4 Security

The proposed ECU architecture integrates confidentiality, integrity, and authentication in automotive CPS. The CM provides three security services: message confidentiality, message integrity, and ECU authentication. We leverage AES-128 (128-bit) encryption to provide confidentiality and SHA-3 based HMAC for authentication and message integrity. Eq. (3) gives the time taken to embed the security primitive to the CAN FD message at the sender node during the non-faulty state of operation.

$$\mathcal{T}^{\mathcal{S}}_{(\mathcal{M}\|\mathcal{C})} = \mathcal{T}^{S}_{cmp} + \mathcal{T}^{S}_{SCV} + \mathcal{T}^{S}_{SCFH} + \bigvee\left(\mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}, \mathcal{T}_{E[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_2]}\right), \quad (3)$$

where $\mathcal{C}$ represents the 64-bit counter, $(\mathcal{M}\|\mathcal{C})$ represents the concatenation of message and the 64-bit counter, $\bigvee(.,.)$ represents a function that selects larger of the two execution times, $\mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}$ represents the time to compute HMAC of message $(\mathcal{M}\|\mathcal{C})$, $\mathcal{T}_{E[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_2]}$ designates the time to compute AES of message $(\mathcal{M}\|\mathcal{C})$, $\mathcal{T}^{S}_{cmp}$ designates the time to compare the outputs of redundant AES and HAMC computation modules at the sender node, $\mathcal{T}^{S}_{SCV}$ represents the time taken by the SCV module to perform the voting decision at the sender node, and $\mathcal{T}^{S}_{SCFH}$ designates the time required by SCFH at the sender node to check the voting decision of SCV and to format the CAN FD message so that the message is ready to be sent to the AP/RTP.

The worst case time for embedding the security primitives to the CAN FD message is given by Eq. (4). The worst case time is the sum of the time to detect the error(s) in computation plus the time to recompute the results using spare modules plus the time to locate the fault. The time to detect error(s) in computation is given by Eq. (3). The time to recompute the results using spare modules is given by $\bigvee(.,.)$ term in Eq. (4) and time to locate the fault is the execution time of SCV and SCFH.

$$\mathcal{T}^{\mathcal{S}_{WC}}_{(\mathcal{M}\|\mathcal{C})} = \mathcal{T}^{\mathcal{S}}_{(\mathcal{M}\|\mathcal{C})} + \mathcal{T}^{S}_{SCFH^*} + \bigvee\left(\mathcal{T}^{*}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}, \mathcal{T}^{*}_{E[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_2]}\right), \quad (4)$$

where $\mathcal{T}^{\mathcal{S}_{WC}}_{(\mathcal{M}\|\mathcal{C})}$ represents the worst case time to integrate the security primitives in $(\mathcal{M}\|\mathcal{C})$, $\mathcal{T}^{\mathcal{S}}_{(\mathcal{M}\|\mathcal{C})}$ represents the time required to incorporate security primitives during normal operation (i.e., without invocation of spare modules) which can be considered as the time to detect the computation error that is uncorrectable by recomputations in the DMR modules, and $\mathcal{T}^{S}_{SCFH^*}$ designates the time required by the SCFH module to identify the faulty module and to activate the reconfiguration subsystem at the sending node.

At the receiver node, first, the CAN FD message is formatted to separate the AES ciphertext and the HMAC. Then, the ciphertext is decrypted by the AES decryption module to generate the original message. The original message (obtained from decryption of the ciphertext) is sent to the HMAC computing module and to the comparators via the input interface. The three comparators independently compare the AES decryption results and the outputs of the comparators are fed to the SCV, which then informs its voting decision to the SCFH. During the operation of the comparators and the SCV, the SHA-3 based HMAC module generates the local HMAC in parallel. Since the comparators and the SCV are faster than the SHA-3 based HMAC module by orders of magnitude, correctness check for AES decryption, and local HMAC calculation can be done in parallel without any conflict. However, this parallel operation incurs some additional signalling overhead on the SCFH. If there is no error in AES decryption, then correctness of local HMAC is assessed after the local HMAC calculation. The FT operation to heal the faulty module is similar at both the sender and the receiver nodes.

Eq. (5) represents the time needed by the receiver to recover the original message with integrity checking. This time is the sum of the time to format the received CAN FD message plus the time to generate the original message sent by the sender node via decryption plus the time to check the integrity of the message by recomputing the HMAC of the recovered original message and comparing the received HMAC with the computed HMAC.

$$\mathcal{T}^{\mathcal{R}}_{(\mathcal{CT}\|\mathcal{MAC})} = \mathcal{T}_{format} + \mathcal{T}_{D[\mathcal{CT},\mathcal{K}_2]} + \mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]} + \mathcal{T}^{R}_{cmp} + \mathcal{T}^{R}_{SCV} + \mathcal{T}^{R}_{SCFH}, \quad (5)$$

where $\mathcal{T}_{format}$ is the time to separate the received CAN FD message into the ciphertext $\mathcal{CT}$ and $\mathcal{MAC}$, $\mathcal{T}_{D[\mathcal{CT},\mathcal{K}_2]}$ is the time to decrypt the ciphertext $\mathcal{CT}$ using secret key $\mathcal{K}_2$; $\mathcal{T}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}$ designates the time to compute the HMAC of the decrypted ciphertext; and $\mathcal{T}^{R}_{cmp}$, $\mathcal{T}^{R}_{SCV}$, and $\mathcal{T}^{R}_{SCFH}$ represent the time taken by comparators, SCV, and SCFH, respectively, at the receiver node.

The worst case time to recover the original message with integrity checking at the receiver node $\mathcal{T}^{\mathcal{R}_{WC}}_{(\mathcal{CT}\|\mathcal{MAC})}$ is given by Eq. (6).

$$\mathcal{T}^{\mathcal{R}_{WC}}_{(\mathcal{CT}\|\mathcal{MAC})} = \mathcal{T}^{\mathcal{R}}_{(\mathcal{CT}\|\mathcal{MAC})} + \mathcal{T}^{*}_{D[\mathcal{CT},\mathcal{K}_2]} + \mathcal{T}^{*}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]} + \mathcal{T}^{R}_{SCFH^*}, \quad (6)$$

where $\mathcal{T}^{\mathcal{R}}_{(\mathcal{CT}\|\mathcal{MAC})}$ represents the time required by the receiver to recover the original message with integrity checking during normal operation (i.e., without invocation of the spare modules), which can be considered equivalent to the time to detect the computation error that is uncorrectable by recomputations in the DMR modules, $\mathcal{T}^{*}_{D[\mathcal{CT},\mathcal{K}_2]}$ is the time to compute AES decryption by the spare module after the error is detected, $\mathcal{T}^{*}_{HMAC[(\mathcal{M}\|\mathcal{C}),\mathcal{K}_1]}$ is the time to compute HMAC from the decrypted ciphertext by the spare module, and $\mathcal{T}^{R}_{SCFH^*}$ is the time taken by the SCFH module to identify the faulty module and to activate the reconfiguration subsystem at the receiver node.

Our proposed approach assumes that the initial AES and HMAC keys are stored in secure tamper resistant memories of ECUs by OEMs. Moreover, these keys are refreshed deterministically over time by the participating ECUs. Our proposed ECU architecture and the security approach provides resilience against the security threat model described in Section 4.1. Our security approach is resilient to *threat 1* because a passive adversary may eavesdrop on traffic but the adversary cannot decrypt the messages without the knowledge of the secret key. Our approach is resilient against *threat 2* because SHA-3 based HMAC with a 128-bit counter value embedded in the ciphertext prevents from the insertion of forged messages, prohibits message modification, and also precludes masquerade and replay attacks. This is possible because there are no known brute-force and analytical attacks against AES and SHA-3 based HMAC computations.

6 Steer-by-Wire System

A SBW system replaces heavy mechanical steering column with an electronic system, which reduces the vehicle weight and eliminates the risk of steering column entering into the cockpit in the event of a car crash. However, these benefits come with the stringent real-time performance and reliability requirements for SBW system. This section elaborates a SBW system that we use as a case study for this work.

6.1 Steer-by-Wire Operational Architecture

In order to successfully replace the conventional steering column, a SBW system needs to provide two main services: front axle control (FAC) and hand-wheel (HW) force feedback (HWF). The SBW architecture used in our study is depicted in Fig. 4. For this study, we focus only on the front axle control part to compute the response time and error resilience of our proposed approaches (illustrated in Section 4). The FT in the SBW system is furnished via redundancy at ECU-, sensor-, and actuator-level. The sensors are connected to ECUs via point-to-point links while ECU-to-ECU communication is accomplished through CAN FD bus. The operation of the SBW system is similar as in our prior work [5], hence we omit the details here for brevity. However, the SBW system uses multicore ECUs for BD and OBD, and uses our proposed ECU architecture for EAF. Furthermore, the ECU-to-ECU communication is carried out via CAN FD bus instead of CAN bus to enable high-speed data transfer.

6.2 QoS and Behavioral Reliability

The delay between the driver's request at HW and the response at FAA has significant impact on the reliability of SBW system. This end-to-end delay/response time ($\mathcal{T}_{res}$) is regarded as a performance (QoS) measure, however, this response time also becomes a reliability measure that impacts safety and availability if this time exceeds a critical threshold value, $\mathcal{T}_{max}$. This $\mathcal{T}_{max}$ is determined by automotive OEMs. The probability that the worst case

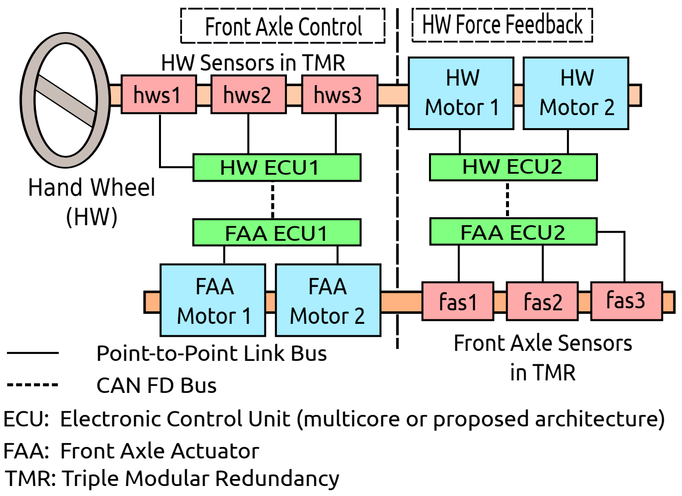

Fig. 4: SBW operational architecture.

response time is less than the critical threshold is termed as *behavioural reliability*. The vehicle's performance and stability is measured in terms of a QoS score, S, and there exists a linear relationship between S and $\mathcal{T}_{res}$ for instantaneous rotation of HW. According to Wilwert et al. [4], for a minimum tolerable S of 11.13, the critical limit $\mathcal{T}_{max}$ for the response time is 11.5 ms, beyond which the vehicle becomes unstable and the safety of driver can be at risk.

In the following, we analytically model the response time and the error resilience provided by our proposed approaches (elaborated in Section 4) for the SBW system subject to the timing constraints imposed by the critical threshold. We consider the FAC part of the SBW system for our analytical modeling. The end-to-end delay/response time is modeled as the sum of pure delay ($\mathcal{D}_p$), mechatronic delay ($\mathcal{D}_{mech}$), and sensing delay ($\mathcal{D}_{sens}$), that is,

$$\mathcal{T}_{res} = \mathcal{D}_p + \mathcal{D}_{mech} + \mathcal{D}_{sens}. \tag{7}$$

The pure delay comprises of ECUs' computational delay for processing the control algorithm and the transmission delay to send the messages from the sending node to the receiving node including bus arbitration. For our secure and dependable architecture, pure delay also includes the computational delay of executing the security and dependability primitives. Since $\mathcal{D}_{mech}$ and $\mathcal{D}_{sens}$ can be upper bounded by a constant value (3.5 ms [38]), we focus on pure delay for our reliability and error resilience analysis. The pure delay has the critical limit $\mathcal{D}_p^{max}$ of 8 ms corresponding to $\mathcal{T}_{max}$ of 11.5 ms. The behavioural reliability can be modeled as $\mathcal{P}_{BR} = \mathcal{P}[\mathcal{D}_p^{wc} < \mathcal{D}_p^{max}]$, where $\mathcal{D}_p^{wc}$ is the worst case $\mathcal{D}_p$ and $\mathcal{P}_{BR}$ is the behavioural reliability. The pure delay for the FAC function can be written as

$$\mathcal{D}_p^{FAC} = \mathcal{D}_{hw}^{ecu1} + \mathcal{D}_{can\_fd}^{channel} + \mathcal{D}_{faa}^{ecu1}, \tag{8}$$

where $\mathcal{D}_{hw}^{ecu1}$ and $\mathcal{D}_{faa}^{ecu1}$ represent the computation time at HW ECU1 and FAA ECU1, respectively, and $\mathcal{D}_{can\_fd}^{channel}$ denotes the time that CAN FD bus takes to transport a message from HW ECU1 to FAA ECU1. The worst case pure delay $\mathcal{D}_p^{wc}$ can be modeled as

$$\begin{aligned}\mathcal{D}_p^{wc} \quad &= \quad rcc1 \cdot \mathcal{D}_{hw}^{ecu1} + rtc \cdot \mathcal{D}_{can\_fd}^{channel} + rcc2 \cdot \mathcal{D}_{faa}^{ecu1}, \\ &\forall \; rcc1, rcc2, rtc \in \mathbb{Z}^{+}, \end{aligned} \tag{9}$$

where $rcc1$ and $rcc2$ represent the number of computations (including recomputations) bounded by NUM_SOFT_ERR threshold (plus the computations done by the spare modules in case of EAF) (refer Algorithm 1) that need to be done at HW ECU1 and FAA ECU1, respectively, to yield an error-free result, rtc represents the number of transmissions (including retransmissions) that needs to be done for error-free sending of a secure message over CAN FD bus, and $\mathbb{Z}^+$ denotes the set of positive integers. Eq. (9) helps to analyze the number of computations (including recomputations) and transmissions (including retransmissions) that are allowed within the real-time budget of automotive CPS application, that is, the constraint $\mathcal{D}_p^{wc} \leq \mathcal{D}_p^{max}$ needs to be satisfied to ensure that the real-time constraints of the automotive CPS application are not violated. We have used this constraint ($\mathcal{D}_p^{wc} \leq \mathcal{D}_p^{max}$) to determine the number of maximum tolerable errors in Section 7.

7 Results

In this section, we present our experimental setup and evaluation results comprising of timing analysis, energy analysis, QoS and behavioral reliability, and feasibility analysis.

7.1 Experimental Setup

Multicore-based ECU Design Implementation: We have implemented both the BD and the OBD (Section 4) on *NXP quad-core iMX6Q SABRE* development board [39], which has *ARM Cortex-A9* CPU core. The 32-bit *Cortex-A9* processor runs *Ubuntu 14.04.4 LTS* at 396 MHz clock speed. The security primitives are coded in C. *OpenMP* is used to provide RMT-based FT on the multicore architecture. The OBD exploits efficient coding strategies for 32-bit ARM platform, such as loop-unrolling, cache-aware programming, alignment of data structures to cache line boundaries in memory, and the use of 32-bit data type only to attain better performance.

Proposed ECU Design Implementation: Our proposed ECU architecture has both the AP/RTP and the PLF. We have implemented the security primitives in the PLF, which is realized by *Xilinx automotive grade Spartan-6 FPGA* [40]. The proposed ECU architecture (Fig. 3) is coded in *Verilog HDL* using *Xilinx ISE 14.7* and the functional verification is done using *ModelSim*. The *post-place and route simulation model* is generated using *Xilinx ISE 14.7*. We have run the simulation model in *ModelSim* to get the execution times. We have used this procedure to get an accurate estimation of the real-world execution time on the board. The total power consumption (both static and dynamic) is obtained via *XPower Analyzer* that comes with the *Xilinx ISE 14.7*. We have used the power and execution time values to compute the energy consumption of the EAF.

Vector CANoe based Setup: We have simulated the SBW system (Fig. 4) in *Vector CANoe 8.5* [41] with CAN FD bus set to 48-byte payload, 1 $Mbps$ arbitration-phase baud rate, and 3 $Mbps$ data-phase baud rate. We have used CAPL (CAN Access Programming Language) to implement the SBW functions on ECUs. Since we have also compared the performance of our proposed secure and dependable approach over other in-vehicle networks (CAN and FlexRay), we have set the following parameters for CAN and FlexRay [42]: CAN settings: baud rate = 1 Mbps, payload size = 8-bytes; FlexRay setting: mixed mode of operation, baud rate = 10 Mbps, payload size = 254-byte. We have used CAPL [43] (CAN Access Programming Language) to implement the SBW functions on ECUs.

Operational Parameters: For the SBW system, we have assumed the steering wheel sensor sampling rate to be fixed at 420 Hz, that is, $\mathcal{D}_{sens} = 2.38\ ms$ (an estimate of $\mathcal{D}_{sens}$ since $\mathcal{D}_{mech}$ plus $\mathcal{D}_{sens}$ can be upper bounded by 3.5 ms [38]). For multicore based SBW system, the ECU operates at 396 MHz clock. The operational current is calculated as 36 mA and the operational voltage as 1.42 V. For our proposed ECU architecture, the PLF, *XA Spartan-6 FPGA*, operates at 50 MHz clock while running the CM.

7.2 Evaluation Results

Timing Analysis: Table 2 depicts the timing performance of BD, OBD, and EAF. The results show that for NFT operational mode, OBD is 1.36× and 1.27× faster than BD at the sender and the receiver nodes, respectively. Results reveal that for FT-RMT, OBD has a speedup of 1.98× and 1.97× over BD at the sender and the receiver nodes, respectively. Finally, for FT-RMT-QED mode, OBD attains a speedup of 1.88× and 1.67× over BD at the sender and the receiver nodes, respectively.

Comparison of BD and EAF reveals that NFT EAF is 52.45× and 26.11× faster than NFT BD at the sender and the receiver nodes, respectively. Furthermore, after embedding FT in BD by FT-RMT and in FPGA by FT-SR-DMR (fault tolerance using self-reconfiguration in DMR), EAF is 62.94× and 37.72× superior than BD at the sender and the receiver nodes, respectively. Lastly, EAF with FT-SR-DMR provides a speedup of 90.19× and 46.75× over BD in FT-RMT-QED mode at the sender and the receiver nodes, respectively.

Comparison of EAF and OBD shows that NFT EAF is faster than NFT OBD by 38.57× and 20.44× at the sender and the receiver nodes, respectively. Moreover, FT-SR-DMR in EAF surpasses FT-RMT in OBD by 31.69× and 19.1× at the sender and the receiver nodes, respectively. Furthermore, FT-SR-DMR in EAF attains a speedup of 47.93× and 27.94× over OBD with FT-RMT-QED at the sender and the receiver nodes, respectively.

We also compare the timing overhead of FT techniques for OBD and EAF. Results indicate that FT-RMT and FT-RMT-QED have time overheads of 10.33% and 61.41%, respectively, over NFT for OBD at the receiver node. Results also show that FT-RMT-QED has a time overhead of 46.3% over FT-RMT for OBD at the receiver node, which results from the insertion of additional check instructions at different points in the program to enable early detection of errors. Results also indicate that FT-SR-DMR incurs a time overhead of 18.11% over NFT for EAF at the receiver node. We observe similar timing overheads at the sending node. The overheads incurred by FT techniques over NFT are inevitable as these reflect the price for incorporating FT in designs.

Energy Analysis: Table 2 and Fig. 5 depict the energy consumption of BD, OBD, and EAF. Results reveal that NFT OBD consumes 1.35× and 1.27× lesser energy than NFT BD at the sender and the receiver nodes, respectively.

TABLE 2: Performance and energy results for BD, OBD, and EAF.

CAN FD Node	Operational Mode	Baseline Design (BD)			Optimized Baseline Design Implementation (OBD)			Proposed ECU FPGA Implementation (EAF)		
		FT Mode	Time (μs)	Energy (μJ)	FT Mode	Time (μs)	Energy (μJ)	FT Mode	Time (μs)	Energy (μJ)
Sender Node	NFT	x	257	13.137	x	189	9.661	x	4.90	2.170
	FT	FT-RMT	411	21.010	FT-RMT	207	10.581	FT-SR-DMR	6.53	6.647
		FT-RMT-QED	589	30.109	FT-RMT-QED	313	16.000			
Receiver Node	NFT	x	235	12.013	x	184	9.406	x	9.00	3.996
	FT	FT-RMT	401	20.499	FT-RMT	203	10.377	FT-SR-DMR	10.63	10.831
		FT-RMT-QED	497	25.406	FT-RMT-QED	297	15.182			

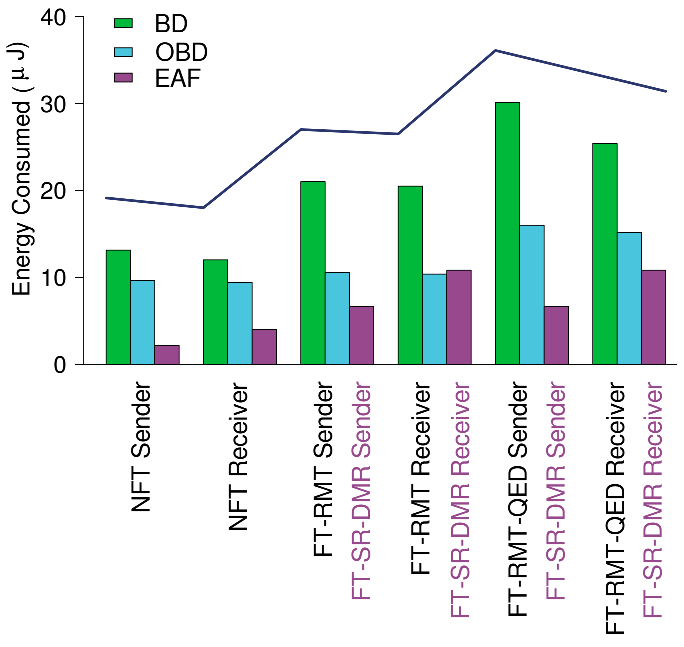

Fig. 5: Energy consumption in BD, OBD, and EAF implementations.

The FT-RMT based OBD is 1.98× more energy-efficient than BD whereas FT-RMT-QED based OBD is 1.88× more energy efficient than BD at the sender node. At the receive node, OBD consumes 1.97× and 1.67× lesser energy than BD for FT-RMT and FT-RMT-QED, respectively. These energy savings ensue from the modification of the security architecture and code optimization for the security primitives for 32-bit ARM platform.

The comparison between EAF and BD reveals that NFT EAF is 6.05× and 3× more energy efficient than NFT BD at the sender and the receiver nodes, respectively. At the sender node, EAF with FT-SR-DMR is 3.16× more energy efficient than BD with FT-RMT and 4.52× more energy efficient than BD with FT-RMT-QED. Similarly, at the receiver node, EAF with FT-SR-DMR is 1.89× more energy efficient than BD with FT-RMT, and 2.34× more energy efficient than BD with FT-RMT-QED.

The comparison between EAF and OBD divulges that NFT EAF results in 4.45× and 2.35× more energy savings than NFT OBD at the sender and the receiver nodes, respectively. Additionally, at the sender node, EAF with FT-SR-DMR engenders 1.59× more energy savings than OBD with FT-RMT, and 2.4× more energy savings than OBD with FT-RMT-QED, respectively. At the receiver node, EAF with FT-SR-DMR consumes 1.04× more energy than

TABLE 3: The maximum number of allowed recomputations and retransmissions on CAN FD bus to yield correct result for the FAC function during faults.

Baseline Design Implementation				
rtc	$rcc1$ with $rcc2 = 1$		$rcc2$ with $rcc1 = 1$	
	FT-RMT	FT-RMT-QED	FT-RMT	FT-RMT-QED
1	18	12	18	14
2	17	12	18	14
5	17	11	17	13
10	15	10	15	12

Optimized Baseline Design Implementation				
rtc	$rcc1$ with $rcc2 = 1$		$rcc2$ with $rcc1 = 1$	
	FT-RMT	FT-RMT-QED	FT-RMT	FT-RMT-QED
1	37	24	37	25
2	36	23	37	25
5	34	22	35	23
10	31	20	32	21

OBD with FT-RMT. This is because that FT-SR-DMR uses redundant modules for FT, which increases the static power consumption of EAF, and thus the energy savings due to faster execution in EAF with FT-SR-DMR as compared to OBD with FT-RMT at the receiver node fail to overcome the effect of increased static power consumption of EAF in this case.

We also compare the energy overhead of FT techniques for OBD and EAF. Results indicate that FT-RMT and FT-RMT-QED have energy overheads of 10.32% and 61.41%, respectively, over NFT for OBD at the receiver node. Results also show that FT-RMT-QED has an energy overhead of 46.3% over FT-RMT for OBD at the receiver node. Results also indicate that FT-SR-DMR incurs an energy overhead of 171.05% over NFT for EAF at the receiver node. This particularly larger energy overhead of FT-SR-DMR over NFT is due to the additional static power consumed by the redundant modules incorporated for providing FT in EAF.

QoS and Behavioral Reliability: We conduct experiments to determine the maximum number of allowable recomputations at SBW ECUs to yield error-free results subject to the critical pure delay $\mathcal{D}_p^{max} = 8\ ms$ and $\mathcal{D}_{can\_fd}^{channel} = 0.118\ ms$. The channel delay is obtained from Vector CANoe simulations [41]. The number of allowable recomputations represents the number of faults (soft errors in this context) the ECU can tolerate. Table 3, which is obtained using Eq. (9) and the behavioral reliability constraint $\mathcal{D}_p^{wc} \leq \mathcal{D}_p^{max}$, depicts $rcc1$, $rcc2$, and rtc permissible for the FAC function during faults subject to the real-time constraints. We have incorporated rtc because CAN and CAN FD buses do not provide FT, and hence

TABLE 4: Pure delay (in ms) for BD, OBD, and EAF.

Operational Mode	BD	OBD	EAF
NFT	0.610	0.491	0.131
FT-RMT/FT-SR-DMR	0.930	0.528	0.135
FT-RMT-QED/FT-SR-DMR	1.204	0.728	0.135

ECUs must retransmit the message(s) in case of transmission errors on the bus for correct and safe operation of the SBW system. We note that the number of faults tolerated at HW ECU1 and FAA ECU1 are given by $(rcc1 - 1)$ and $(rcc2 - 1)$, respectively, since even in absence of error(s), HW ECU1 and FAA ECU1 still require one computational run time for the FAC function.

Table 3 depicts the number of permissible recomputations on ECUs and retransmissions on CAN FD bus to yield an error-free result for the FAC function during faults for both BD and OBD. Results indicate that OBD can tolerate up to $(rcc1 - 1) + (rcc2 - 1) = 71$ faults with one transmission error (i.e., $rtc = 2$) without violating $\mathcal{D}_p^{max}$ critical limit for FT-RMT technique. Comparison between BD and OBD reveals that OBD can tolerate 113% (2.13×) and 94% (1.94×) more faults on average than BD for FT-RMT and FT-RMT-QED, respectively. The table also shows the number of allowable retransmissions under the $\mathcal{D}_p^{max} = 8\ ms$ constraint. Results indicate that OBD permits 16.19× and 29.81× more retransmissions on average than BD for FT-RMT and FT-RMT-QED operation mode, respectively, for the same critical pure delay because of the lesser ECU execution time for security and dependability primitives for OBD than BD.

Feasibility Analysis over CAN FD: Table 4 and Table 5 depict the pure delay and response time for CAN FD channel (defined and formulated in Section 6.2) for BD, OBD, and EAF. Results verify that the pure delay for all the three implementations (BD, OBD, and EAF) are well within the critical threshold limit of pure delay $\mathcal{D}_p^{max} = 8\ ms$. Additionally, the response time for the three implementations also satisfies the critical response time threshold of $\mathcal{T}_{max} = 11.5\ ms$. Results indicate that the pure delay ($\mathcal{D}_p$) and the worst-case pure delay ($\mathcal{D}_p^{wc}$) for EAF are $0.135\ ms$ and $0.152\ ms$, respectively. Results show that EAF provides improvements of 8.9× and 5.4× in pure delay over BD and OBD (with FT-RMT-QED), respectively, with CAN FD as in-vehicle bus. Results further shown that the EAF provides improvements of 1.3× and 1.2× in response time over BD and OBD (with FT-RMT-QED), respectively, with CAN FD as in-vehicle bus. These results verify the feasibility of our proposed secure and dependable approach and BD, OBD, and EAF for automotive CPS with CAN FD as in-vehicle bus for safety-critical applications.

From Table 4, we observe that the pure delay for OBD is 1.5× lesser than BD on average. Further, on average, the pure delay for EAF is 4.3× and 6.8× lesser than OBD and BD, respectively. Comparing the response times, we observe that OBD has 9.81% lesser response time than BD on average. Moreover, EAF response time is on average 12.34% and 21.49% lesser than that of OBD and BD, respectively. The results indicate that our proposed secure and FT ECU design (EAF) permits much more time for control processing of the implemented automotive function (e.g., SBW) as compared to the multicore-based implementations (BD and OBD).

TABLE 5: End-to-end delay or response time (in ms) for BD, OBD, and EAF for CAN FD channel delay of $0.118\ ms$ and $\mathcal{D}_{mech} + \mathcal{D}_{sens} = 3.5\ ms$.

Operational Mode	BD	OBD	EAF
NFT	4.110	3.991	3.631
FT-RMT/FT-SR-DMR	4.430	4.028	3.635
FT-RMT-QED/FT-SR-DMR	4.704	4.228	3.635

Comparison of the Proposed Approaches For Different In-Vehicle Networks: We have performed experiments to determine the impact of using different in-vehicle buses (CAN, CAN FD, and FlexRay) on the end-to-end delay and response time of BD, OBD, and EAF. Table 6 depicts the response time of the SBW system (Fig. 4) when using different in-vehicle buses in combination with different ECU architectures (BD, OBD, and EAF). Results verify that the pure delay as well as response time for our secure and dependable approach leveraging BD, OBD, and EAF satisfy the critical pure delay of $\mathcal{D}_p^{max} = 8\ ms$ and critical response time of $\mathcal{T}_{max} = 11.5\ ms$ for all three in-vehicle networks (CAN, CAN FD, and FlexRay). These results demonstrate the feasibility and scalability of the proposed approach and architectures for different in-vehicle networks.

Results indicate that CAN FD and FlexRay are better alternatives to traditional CAN bus as they provide higher bandwidths and lower latencies. In order to transport the *48-bytes payload* as required by our proposed secure and dependable approach (elaborated in Section 4), CAN FD requires 37× lesser time than CAN. We derive this improvement in transmission time furnished by CAN FD over CAN as follows. Since six CAN messages are required to transmit the encrypted message and hash (48-bytes payload), the time required for transmitting six CAN messages is equal to $0.74\ ms \times 6 = 4.44\ ms$, whereas CAN FD requires only one message to transfer 48-bytes payload (the maximum payload size of the CAN FD message is 64 bytes [44]) with a transfer time of $0.12\ ms$. Hence, CAN takes $4.44/0.12 = 37\times$ more time than CAN FD to transmit the 48-bytes payload required by our proposed secure and dependable approach. Similarly, FlexRay requires only one message to transfer 48-byte payload (the maximum payload size of the FlexRay message is 254 bytes [45]) with a transfer time of $0.05\ ms$. Hence, CAN takes $4.44/0.05 = 88.8\times$ more time than FlexRay to transmit the 48-bytes payload required by our proposed secure and dependable approach. Moreover, FlexRay requires $0.12/0.05 = 2.4\times$ lesser time than CAN FD to transmit the 48-bytes payload. Furthermore, FlexRay offers FT in communication as it provides redundant communication channels.

Comparison between pure delays indicate that EAF with FT-SR-DMR decreases the pure delay by 19.34%, 88.64%, and 94.10% over BD with FT-RMT-QED for CAN, CAN FD, and FlexRay, respectively. Similarly, EAF with FT-SR-DMR decreases the pure delay by 11.74%, 81.23%, and 89.85% over OBD with FT-RMT-QED for CAN, CAN FD, and FlexRay, respectively. Comparison between response times indicate that EAF with FT-SR-DMR decreases the response time by 6.94%, 14.02%, and 14.25% over OBD with FT-RMT-QED for CAN, CAN FD, and FlexRay, respectively. For CAN, pure delay and response time are dominated by

TABLE 6: Response time $\mathcal{T}_{res}$ and pure delay $\mathcal{D}_p$ (in ms) of the SBW subsystem for different in-vehicle buses (CAN, CAN FD, and FlexRay) assuming $\mathcal{D}_{mech} + \mathcal{D}_{sens} = 3.5\ ms$.

CAN FD Node	Operational Mode	Baseline Design (BD)			Optimized Baseline Design Implementation (OBD)			Proposed ECU FPGA Implementation (EAF)		
		FT Mode	$\mathcal{D}_p$ (ms)	$\mathcal{T}_{res}$ (ms)	FT Mode	$\mathcal{D}_p$ (ms)	$\mathcal{T}_{res}$ (ms)	FT Mode	$\mathcal{D}_p$ (ms)	$\mathcal{T}_{res}$ (ms)
CAN	NFT	x	4.932	8.432	x	4.813	8.313	x	4.454	7.954
Latency = 0.74 ms	FT	FT-RMT	5.252	8.752	FT-RMT	4.850	8.350	FT-SR-DMR	4.457	7.957
		FT-RMT-QED	5.526	9.026	FT-RMT-QED	5.050	8.550			
CAN FD	NFT	x	0.612	4.112	x	0.493	3.993	x	0.134	3.634
Latency = 0.12 ms	FT	FT-RMT	0.932	4.432	FT-RMT	0.530	4.030	FT-SR-DMR	0.137	3.637
		FT-RMT-QED	1.206	4.706	FT-RMT-QED	0.730	4.230			
FlexRay	NFT	x	0.542	4.042	x	0.423	3.923	x	0.064	3.564
Latency = 0.05 ms	FT	FT-RMT	0.862	4.362	FT-RMT	0.460	3.960	FT-SR-DMR	0.067	3.567
		FT-RMT-QED	1.136	4.636	FT-RMT-QED	0.660	4.160			

transmission time, which explain the lower improvements in pure delay and response time by EAF over BD and OBD in case of CAN bus. The improvements in response time and pure delay rendered by EAF over BD and OBD become more pronounced for CAN FD and FlexRay as computation times becomes the more significant fraction than transmission time in pure delay as well as response time. The improvements in pure delay and response time furnished by EAF over BD and OBD are due to the lower execution time of the security and dependability primitives in EAF as compared to BD and OBD.

8 Future Research Directions

Although security and dependability of automotive CPS has gained much interest in both academia and industry in recent years, there still exist various research challenges in this domain that demand attention. In this section, we highlight these challenges and future research directions that need consideration to realize next generation of secure and dependable automobiles.

Reconfigurability in Automotive CPS Design: Although this paper proposes a reconfigurable ECU architecture that simultaneously integrates security and dependability primitives in automotive CPS, further research is needed to explore the potential of utilizing reconfigurability in automotive CPS. For instance, the proposed approach in this work (Fig. 1) can be extended to embed configurability in security and reliability parameters in order to ensure meeting real-time requirements based on changing application requirements and environmental stimuli (e.g., transient fault rate and in-vehicle bus load). The security and reliability parameters that can be adapted include determining N value for NMR (N modular redundancy) in an automotive processor that permits using multiple (N) cores or resources for providing FT, number of comparison points for QED, and key lengths for AES encryption/decryption and HMAC.

In-Vehicle Networks: Modern automobiles utilize a variety of in-vehicle networks, such as LIN (Local Interconnect Network), MOST (Media Oriented Systems Transport), CAN, CAN FD, and FlexRay, for different applications. LIN is typically used for automotive body electronics including air conditioning systems, seats, doors, climate control, intelligent windshield wipers, and sunroof actuators. MOST is utilized for multimedia and entertainment applications. CAN has been used traditionally for real-time and safety-critical application, such as engine control, transmission, and anti-lock braking braking/ABS. Recently, CAN FD and FlexRay are being considered for automotive applications requiring higher bandwidth than CAN. More recently, automotive Ethernet has beginning to permeate in modern automobiles for infotainment and high-bandwidth applications. However, further research is needed in in-vehicle networks, in particular, CAN FD, FlexRay, and Ethernet, to help determine the most appropriate network for different safety-critical applications in modern automotive CPS and in autonomous vehicles.

Secure Storage of Secret Keys: Most of the contemporary security methods (discussed in Section 3.1) for automotive CPS are based on secret keys that are to be stored in automotive ECUs. The leakage of these secret keys can forfeit the security of the entire automotive system. The secret keys can be stored in secure tamper-resistant memories (as assumed in this paper), however, a large number of attack vectors, such as side-channel attacks, fault injection attacks, microprobing, reverse engineering, and software attacks have been developed for estimation, cloning, and extraction of secret keys stored in nonvolatile memory. For instance, cryogenic memory attacks are possible even if TPM is in place as illustrated in [46] [47] and summarized here. Dynamic random access memory (DRAM), also used in modern ECUs, can preserve their contents and memory images for several seconds after power is lost, even at room temperature and even if removed from the automotive bus. It has been shown by researchers that cryogenically frozen DRAM can retain the data for several minutes to an hour. Hence, cold (cryogenic) reboots can be used to mount successful attacks for extracting cryptographic keys from memory images [47]. Consequently, further research is needed to ensure secure storage of secret keys in automotive ECUs.

Secure Generation and Distribution of Secret Keys: To alleviate the perils of secret keys' storage, hardware-based security techniques such as physically unclonable functions (PUFs) can be utilized to generate secret keys without the need for storing them in nonvolatile memory. Another relevant challenge is secure distribution of secret keys between automotive ECUs. Asymmetric cryptography is widely used in Internet for secret key distribution. However, resource constraint of automotive ECUs makes it difficult to implement complex secret key exchange protocols with large key lengths required to provide adequate security. Moreover, private keys need to be stored in automotive ECUs for key distribution using asymmetric key distribution, which makes private keys vulnerable to security attacks and extraction. Hence, there is a need to develop security mechanisms and protocols that can solve the problem of key generation and distribution in automotive CPS.

ECU Authentication: Authentication of ECUs is paramount for security of automotive CPS as the authentication ensures that only legitimate ECUs participate in communication over the in-vehicle network. In this work, possession of secure symmetric keys is considered a means for ECU authentication because only authorized ECUs will have the secret symmetric keys. However, other means of ECU authentication needs to be developed. Asymmetric cryptographic techniques can be used for ECU authentication but require public key infrastructure (PKI) support to accomplish that. Asymmetric cryptographic techniques also enable revocation of certificates in case ECUs become compromised or stolen, however, there will always be a period when certificates are invalid but revocation lists have not been communicated to all ECUs especially in case of automobiles which are frequently driven in remote areas with limited Internet and cellular connectivity. Hence, accomplishing ECU authentication without PKI support is a topic of future research.

Artificial Intelligence Safety and Security: Artificial intelligence (AI) has revolutionized the automotive industry by driving the development of increasingly intelligent autonomous vehicles. The automotive AI market is expected to be valued at $11k million by 2025 [48]. AI has become an essential component of automated drive technology by enabling voice recognition, sentiment analysis, image recognition, object detection, motion detection, and machine vision in autonomous vehicles. However, machine learning (ML) algorithms are susceptible to various types of adversarial attacks. *Adversarial examples* are an instance of these adversarial attacks where an adversary introduces small perturbations to the legitimate input to force an ML algorithm to misclassify (misinterpret) while the perturbed input remains correctly classifiable by the human observer [49]. These adversarial examples can cause an autonomous vehicle's ML algorithm to malfunction (e.g., the algorithm may classify a stop sign as an yield sign) and thus put the safety of the vehicle's passengers, pedestrians, and property at risk. Therefore, to ensure the safety and security of autonomous vehicles, further research is needed in AI safety and security to mitigate the vulnerabilities of AI algorithms.

Privacy of Vehicle Owner and Passengers: Next generation of automobiles, including autonomous vehicles, will gather and maintain identifying information about the vehicle owner and passengers for various purposes, such as to authenticate authorized users and to customize safety, comfort, and entertainment settings [50]. The stored information will probably be able to identify owner and passengers and their behavior and activities with a high degree of certainty, and thus can be exploited by attackers to infringe on the privacy of vehicle owner and passengers. The V2X communication in ITS will further exacerbate privacy issues as vehicles will be communicating with each other and the infrastructure, which pose the risk of exposing identity of vehicles and the drivers of those vehicles. To better ensure the privacy of vehicle owners and passengers, automotive systems need to adhere to *privacy by design* approach where privacy is embedded in design specifications and architecture of systems and processes at the outset rather than as an afterthought [51]. The promising approaches to preserve privacy by design include privacy-preserving computing, homomorphic encryption, and blockchain cryptography. However, further research is needed to integrate privacy by design in automotive CPS and thus ensure privacy of vehicle owners and passengers.

9 Conclusions

In this paper, we have proposed a novel ECU architecture for automotive cyber-physical systems (CPS) that simultaneously integrates both security and dependability primitives in the design with negligible performance, energy, and resources overhead. We have implemented our proposed ECU architecture on Xilinx Automotive (XA) Spartan-6 FPGA, which we refer to as EAF. We have also proposed a secure and dependable approach for automotive CPS design that leverages our proposed ECU architecture. We have demonstrate the effectiveness of our proposed architecture and approach using a steer-by-wire (SBW) application over controller area network with flexible data rate (CAN FD) as a case study. We have further optimized and implemented a prior secure and dependable automotive work (baseline design (BD)) on the NXP quad-core iMX6Q SABRE automotive board. We refer to the optimized implementation as optimized baseline design (OBD).

Results reveal that our optimized design can tolerate 113% (2.13×) more faults on average than BD. Results also divulge that our proposed ECU architecture can attain a speedup of 90.19× while consuming 4.52× lesser energy over BD. Furthermore, EAF can attain a speedup of 47.93× while consuming 2.4× lesser energy than OBD. We further perform a comparative analysis of prior designs (BD and OBD) and the proposed ECU architecture (EAF) for different in-vehicle networks, viz., CAN, CAN FD, and FlexRay. Results verify the feasibility as well as the superiority of EAF over BD and OBD in terms of pure delay and response time. Results show that EAF in fault tolerance (FT) mode can reduce the pure delay by 19.34%, 88.64%, and 94.10% over BD with FT for CAN, CAN FD, and FlexRay, respectively. Finally, we have also highlighted future research directions for designing secure and dependable automotive CPS.

Acknowledgments

This work was supported by the National Science Foundation (NSF) (NSF CNS 1743490). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] ISO. (2011, November) ISO 26262 road vehicles - functional safety. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=43464

[2] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. Morgan Kaufmann Publishers, 2007.

[3] A. Munir and F. Koushanfar, "Design and Analysis of Secure and Dependable Automotive CPS: A Steer-by-Wire Case Study," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, June 2018.

[4] C. Wilwert, Y.-Q. Song, F. Simonot-Lion, Loria-Trio, and T. Clément, "Evaluating Quality of Service and Behavioral Reliability of Steer-by-Wire Systems," in *Proc. of IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Lisbon, Portugal, September 2003.

[5] A. Munir and F. Koushanfar, "Design and Performance Analysis of Secure and Dependable Cybercars: A Steer-by-Wire Case Study," in *Proc. of IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, Nevada, January 2016.

[6] A. Vinel, N. Lyamin, and P. Isachenkov, "Modeling of V2V Communications for C-ITS Safety Applications: a CPS Perspective," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1600–1603, August 2018.

[7] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, May 2010, pp. 447–462.

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *Proc. of the 20th USENIX conference on Security (SEC)*, San Francisco, California, August 2011.

[9] M. L. Chavez, C. H. Rosete, and F. R. Henriguez, "Achieving confidentiality security service for can," in *2005 15th International Conference on Electronics, Communication and Computers CONIELECOMP*, Puebla, Mexico, Feb 2005, pp. 166–170.

[10] ISO. (2000) Iso 7498-2:1989: Information processing systems – open systems interconnection – basic reference model – part 2: Security architecture. [Online]. Available: https://www.iso.org/standard/14256.html

[11] B. Groza, S. Murvay, A. van Herrewege, and I. Verbauwhede, *LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks*. Springer Berlin Heidelberg, 2012, pp. 185–200.

[12] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol," in *2012 International Conference on Cyber Security (CyberSecurity)*, Washington, DC, USA, Dec 2012, pp. 1–7.

[13] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *IEEE 68th Vehicular Technology Conference, 2008*, Calgary, BC, Sep 2008, pp. 1–5.

[14] A. V. Herrewege, D. Singelee, and I. Verbauwhede, "CANauth - a simple, backward compatible broadcast authentication protocol for CAN bus," in *ECRYPT Workshop on Lightweight Cryptography*, Louvain-la-Neuve, Belgium, Nov 2011.

[15] T. Ziermann, S. Wildermann, and J. Teich, "CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates." in *2009 Design, Automation Test in Europe Conference Exhibition*, Nice, France, Apr 2009, pp. 1088–1093.

[16] M. Wolf and T. Gendrullis, "Design, implementation, and evaluation of a vehicular hardware security module," in *Proceedings of the 14th International Conference on Information Security and Cryptology*, Seoul, Korea, Nov-Dec 2012, pp. 302–318.

[17] E. Beckschulze, F. Salewski, T. Siegbert, and S. Kowalewski, "Fault handling approaches on dual-core microcontrollers in safety-critical automotive applications," in *2008 International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation ISoLA*, Porto Sani, Greece, Oct 2008.

[18] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-tolerant platforms for automotive safety-critical applications," in *2003 ACM International Conference on Compilers, Architecture and Synthesis for Embedded Systems CASES*, San Jose, California, Oct-Nov 2003.

[19] M. Rebaudengo, M. S. Reorda, M. Torchiano, and M. Violante, "Soft-error detection through software fault-tolerance techniques," in *1999 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems DFT*, Oct 1999, pp. 210–218.

[20] Fujitsu. (2012, Feb) Secure hardware extension. [Online]. Available: https://www.escrypt.com/fileadmin/escrypt/pdf/WEB_Secure_Hardware_Extension_Wiewesiek.pdf

[21] F. SIT. (2008) E-safety vehicle intrusion protected applications. [Online]. Available: http://www.evita-project.org/

[22] PRESERVE. (2015) PRESERVE - preparing secure V2X communication systems. [Online]. Available: https://www.preserve-project.eu/

[23] T. C. Group. (2016) Trusted computing group - open standards for security and technology. [Online]. Available: http://www.trustedcomputinggroup.org/

[24] ARM. (2015) Trustzone - ARM. [Online]. Available: http://www.arm.com/products/processors/technologies/trustzone/

[25] Automotive Electronics Council. (2007, May) Aec-q100-rev-g, failure mechanism based stress test qualification for integrated circuits. [Online]. Available: http://www.aecouncil.com/Documents/AEC_Q100_Rev_G_Base_Document.pdf

[26] CAN in Automation (CiA). (2018, November) Can data link layers. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-data-link-layers/

[27] T. Hong et al., "QED: Quick Error Detection Tests for Effective Post-Silicon Validation," in *IEEE ITC*, Austin, Texas, Nov 2010.

[28] C.-F. Lu, Y.-S. Kao, H.-L. Chiang, and C.-H. Yang, "Fast implementation of AES cryptographic algorithms in smart cards," in *Proceedings of IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003*, Taipei, Taiwan, Oct 2003, pp. 573–579.

[29] A. Javed, "Fast implementation of AES on mobile devices," in *8th International Network Conference*, Heidelberg, Germany, Jul 2010, pp. 133–142.

[30] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin, *Efficient Software Implementation of AES on 32-Bit Platforms*. Springer Berlin Heidelberg, 2004, pp. 159–171.

[31] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. (2016, May) The keccak sponge function family. [Online]. Available: http://keccak.noekeon.org/

[32] K. Atasu, L. Breveglieri, and M. Macchetti, "Efficient AES implementations for ARM based platforms," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, Nicosia, Cyprus, Mar 2004, pp. 841–845.

[33] C. Shore, "Efficient C code for ARM devices," in *ARM Technology Conference 2010*, Santa Clara, California, USA, Sep 2010, pp. 1–14.

[34] Xilinx. (2018, November) Zynq-7000 all programmable soc. [Online]. Available: https://www.xilinx.com/support/documentation/product-briefs/zynq-7000-product-brief.pdf

[35] S. Kundu and S. M. Reddy, "Embedded totally self-checking checkers: A practical design," *IEEE Design Test of Computers*, vol. 7, no. 4, pp. 5–12, Aug 1990.

[36] *Spartan-6 FPGA Configuration User Guide (v2.8)*, Nov 2015.

[37] *LogiCORE IP XPS HWICAP v5.01a Product Specification*, Jun 2011.

[38] K. Klobedanz, C. Kuznik, A. Thuy, and M. Wolfgang, "Timing modeling and analysis for autosar-based software development - a case study," in *IEEE/ACM DATE*, Dresden, Germany, Mar 2010, pp. 642–645.

[39] NXP. (2018, November) Rd-imx6q-sabre: Sabre board for smart devices based on the i.mx 6quad applications processors. [Online]. Available: https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/sabre-development-system/sabre-board-for-smart-devices-based-on-the-i.mx-6quad-applications-processors:RD-IMX6Q-SABRE

[40] Xilinx. (2018, November) Automotive-grade xa spartan-6 fpga family. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/xa-spartan-6.html

[41] VECTOR. (2018, October) Testing ecus and networks with CANoe. [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/canoe/

[42] B. Poudel, N. K. Giri, and A. Munir, "Design and Comparative Evaluation of GPGPU- and FPGA-based MPSoC ECU Architectures for Secure, Dependable, and Real-Time Automotive CPS," in *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Seattle, Washington, July 2017.

[43] *Programming with CAPL*, Dec 2004.

[44] CAN in Automation (CiA). (2018, November) Can fd - the basic idea. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-fd/

[45] AUTOSAR. (2011, April) Specification of flexray transport layer. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/3-2/AUTOSAR_SWS_FlexRay_TP.pdf

[46] ZDNet. (2008, February) Cryogenically frozen ram bypasses all disk encryption methods. [Online]. Available: https://www.zdnet.com/article/cryogenically-frozen-ram-bypasses-all-disk-encryption-methods/

[47] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest We Remember: Cold Boot Attacks on Encryption Keys," in *Proc. of the 17th USENIX conference on Security (SEC)*, San Jose, California, July 2008.

[48] S. Gadam. (2018, April) Artificial intelligence and autonomous vehicles. [Online]. Available: https://medium.com/datadriveninvestor/artificial-intelligence-and-autonomous-vehicles-ae877feb6cd2

[49] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM Asia Conference on Computer and Communications Security*. Abu Dhabi, United Arab Emirates: ACM, April 2017, pp. 506–519.

[50] Norton Rose Fulbright. (2017, July) The privacy implications of autonomous vehicles. [Online]. Available: https://www.dataprotectionreport.com/2017/07/the-privacy-implications-of-autonomous-vehicles/

[51] A. Cavoukian, "Privacy by design: Origins, meaning, and prospects for assuring privacy and trust in the information era," in *Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards*. IGI Global, 2012, pp. 170–208.

Bikash Poudel is currently working at Intel Corporation as a post silicon validation engineer. He received his M.S. in Computer Science and Engineering from the University of Nevada, Reno in 2017. He also worked as a Staff Research Assistant at Kansas State University in 2017. His research interests include hardware security, computer architecture, embedded and cyber-physical systems, and design validation.

Arslan Munir (M'09, SM'17) is currently an Assistant Professor in the Department of Computer Science (CS) at Kansas State University (K-State). He holds a Michelle Munson-Serban Simu Keystone Research Faculty Scholarship from the College of Engineering. He was a postdoctoral research associate in the Electrical and Computer Engineering (ECE) department at Rice University, Houston, Texas, USA from May 2012 to June 2014. He received his M.A.Sc. in ECE from the University of British Columbia (UBC), Vancouver, Canada, in 2007 and his Ph.D. in ECE from the University of Florida (UF), Gainesville, Florida, USA, in 2012. From 2007 to 2008, he worked as a software development engineer at Mentor Graphics in the Embedded Systems Division.

Munir's current research interests include embedded and cyber-physical systems, secure and trustworthy systems, hardware-based security, computer architecture, multicore, parallel computing, distributed computing, reconfigurable computing, artificial intelligence (AI) safety and security, data analytics, and fault tolerance. Munir received many academic awards including the doctoral fellowship from Natural Sciences and Engineering Research Council (NSERC) of Canada. He earned gold medals for best performance in electrical engineering, gold medals and academic roll of honor for securing rank one in pre-engineering provincial examinations (out of approximately 300,000 candidates). He is a Senior Member of IEEE.